Signature File Partitioning (cont.)

- Fixed prefix

Use the initial part as a key

We want to consider both sequential and parallel processing

Partition activation ratio (PAR), signature activation ratio (SAR) (covered last time)

\[
\begin{align*}
S1 &: 0111 1000 \\
S2 &: 1000 1011 \\
S3 &: 0011 1100 \\
S4 &: 1100 0011 \\
S5 &: 0110 1100 \\
S6 &: 1001 0011 \\
S7 &: 0000 1111
\end{align*}
\]

\[
\begin{array}{cccc}
00 & 01 & 10 & 11 \\
S3 & S1 & S2 & S4 \\
S7 & S5 & S6 &
\end{array}
\]

**Q1:** 1110 0001

**Q2:** 0000 1111

**Q3:** 0110 0011

We select the pages of which the prefixes \( P_i \) satisfy the rule \( P_i \land Q_i = Q_i \).

\[
k = 2
\]

**Q1:** 111... 11 (1 page selected)

**Q2:** 000... 00, 01, 10, 11 (4)

**Q3:** 011... 01, 11 (2)

**Sequential processing**

All queries arrive at \( t=0 \)
Parallel processing

Turnaround time = Time of completion – Time of arrival

<table>
<thead>
<tr>
<th>Seq.</th>
<th>Par.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>1</td>
</tr>
<tr>
<td>Q2</td>
<td>1</td>
</tr>
<tr>
<td>Q3</td>
<td>3</td>
</tr>
</tbody>
</table>

Serial average turnaround time = (1+5+7) = 3 tu  (tu: Time units)
Parallel average turnaround time = (1+2+3) = 2 tu

Throughput (T): No. of jobs completed per unit time

\[ T_s = \frac{3}{7} \quad T_p = \frac{3}{3} \quad T_p > T_s \]
Signature Tree Structure

Page signatures are ANDed (“superposed”) to obtain superpage signatures. Any branch that do not satisfy the query condition is simply pruned.

Information Filtering

- Also known as “selective dissemination of information”
- It is the mirror image of information retrieval
- Change the roles of queries and documents

User profiles ≡ Queries

We receive documents, and we match these documents with user profiles. That is, we send the incoming document to the matching user profiles. Therefore the process is reversed.

We can convert the documents to vectors, and calculate similarity

Or, if an incoming document contains all user profile terms, send it to the owner of the profile (we use the AND operand).

There are several approaches described in the paper:

**Brute force approach (Sequential comparison):**

We compare the incoming document with each user profile one by one.

**Counting method:**

Sample profiles:
- \( p_1: (a, b) \)
- \( p_2: (a, d) \)
- \( p_3: (a, d, e) \)
- \( p_4: (b, f) \)
- \( p_5: (c, d, e, f) \)

Sample document: a, c, a, f, b, c
Unique terms: \{a, b, c, f\}

Construct an inverted index:

<table>
<thead>
<tr>
<th>Directory (in memory)</th>
<th>Inverted list (on disk)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>p1, p2, p3</td>
</tr>
<tr>
<td>b</td>
<td>p1, p4</td>
</tr>
<tr>
<td>c</td>
<td>p5</td>
</tr>
<tr>
<td>d</td>
<td>p2, p3, p5</td>
</tr>
<tr>
<td>e</td>
<td>p3, p5</td>
</tr>
<tr>
<td>f</td>
<td>p4, p5</td>
</tr>
</tbody>
</table>

Before processing a document set all count entries equal to 0.

Take the unique terms of the incoming document, visit the posting list and increment the counts for matching profiles.

<table>
<thead>
<tr>
<th></th>
<th>Total</th>
<th>( p_1 )</th>
<th>( p_2 )</th>
<th>( p_3 )</th>
<th>( p_4 )</th>
<th>( p_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>f</td>
<td></td>
</tr>
<tr>
<td>( p_1 )</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p_2 )</td>
<td>2</td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p_3 )</td>
<td>3</td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>( p_4 )</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( p_5 )</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

In this example, we obtained the required counts for \( p_1 \) and \( p_4 \); so they will receive the document.
Key method

Assume that we know the occurrence frequency of the terms in documents:

\[
\begin{array}{ccccccc}
  a & b & c & d & e & f \\
  \uparrow & \uparrow & & & & \\
\end{array}
\]

Least frequent term

Most frequent term

A profile appears in one of the inverted lists only in its term that appears most infrequently in documents:

<table>
<thead>
<tr>
<th>Directory</th>
<th>Inverted list (posting list)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>p₁ 1 b p₂ 1 d p₃ 2 d e</td>
</tr>
<tr>
<td>b</td>
<td>p₄ 1 f</td>
</tr>
<tr>
<td>c</td>
<td>p₅ 3 d e f</td>
</tr>
<tr>
<td>d</td>
<td>We skip these as stopwords</td>
</tr>
<tr>
<td>e</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
</tr>
</tbody>
</table>

The expectation: The term that appears less frequently in documents will appear more frequently in user profiles.

Exercises:

1. Distribute the signatures below into pages with (a) k=2 and (b) k=3 prefix.
   
   S₁ = 0011 1001, S₂ = 1010 1100, S₃ = 0101 0101, S₄ = 1111 0000,
   S₅ = 0001 1110, S₆ = 1101 0100, S₇ = 1001 0011, S₈ = 0110 0110

   a. For k=2, pages have these prefixes: P₀=00, P₁=01, P₂=10, P₃=11.
      Then P₀ = {S₁, S₅}; P₁ = {S₃, S₈}; P₂ = {S₂, S₇}; P₃ = {S₄, S₆}

   b. For k=2, pages are P₀=000, P₁=001, P₂=010, P₃=011, P₄=100, P₅=101, P₆=110, P₇=111.

2. For both cases, which page gets retrieved the most? Why?

   For k=2, P₃ gets retrieved every time. For k=3, P₇ gets retrieved every time. The reason is that
   these prefixes are all 1s, and Pᵢ & Qᵢ = Qᵢ is satisfied for any query.